A Reevaluation and Benchmark of Hidden Markov Models

van Oosten, Jean-Paul; Schomaker, Lambertus

Published in:
14th International Conference on Frontiers in Handwriting Recognition

DOI:
10.1109/ICFHR.2014.95

Publication date:
2014

A reevaluation and benchmark of hidden Markov models

Jean-Paul van Oosten, Lambert Schomaker
Artificial Intelligence
University of Groningen
The Netherlands
{J.P.van.Oosten,L.Schomaker}@ai.rug.nl

***Abstract*—Hidden Markov models are frequently used in handwriting-recognition applications. While a large number of methodological variants have been developed to accommodate different use cases, the core concepts have not been changed much. In this paper, we develop a number of datasets to benchmark our own implementation as well as various other tool kits. We introduce a gradual scale of difficulty that allows comparison of datasets in terms of separability of classes. Two experiments are performed to review the basic HMM functions, especially aimed at evaluating the role of the transition probability matrix. We found that the transition matrix may be far less important than the observation probabilities. Furthermore, the traditional training methods are not always able to find the proper (true) topology of the transition matrix. These findings support the view that the quality of the features may require more attention than the aspect of temporal modelling addressed by HMMs.**

***Keywords*-Hidden Markov Models; State-transition probabilities; Baum-Welch; Benchmark**

I. Introduction

In 1989, Rabiner published the seminal work[1] on hidden Markov models (HMMs), with applications in speech recognition. Since then, HMMs have been used in other domains as well, such as segmenting gene sequences[2] and handwriting recognition[3], [4]. In this paper, we will discuss the applications in this last domain and HMMs in general.

There is a large number of variations of the regular HMMs that Rabiner wrote about, ranging from pseudo 2D-HMMs[5], to truly 2D-HMMs (Markov random fields)[6] and explicit duration modelling[7], to nested HMMs[8] and many more. In the core, these variations are still HMMs, usually trained using the Baum-Welch algorithm. When the data is already labelled with hidden states, however, the transition probability matrix can be modelled directly, without using the potentially more unpredictable EM-based approach. This is the case in segmenting gene sequences with profile HMMs[2] for example, using many pattern heuristics to identify state-transitions in the sequence.

The overall HMM architecture (e.g., determining the number of states, transition matrix topology and integrating it into a larger framework) requires a lot of human effort. However, to our knowledge, no real benchmark has been proposed to test algorithm variants of HMM implementations. In section II, we will discuss how such a benchmark can be constructed. It will not only provide a way to compare results, but also allow one to determine the difficulty of a particular dataset.

The goal of this paper is to investigate the core of HMMs. HMMs consist of three main components: the initial state probability distribution ($\vec{\pi}$), the transition probability matrix ($\mathbf{A}$) and the observation probability functions ($\mathbf{B}$). While the role of the initial state probability distribution is known to be of relatively small importance (especially in left-right topologies such as Bakis, since these models always start in the first state), it is hard to find concrete information on the relative importance of the transition and observation probabilities for optimal performance in the literature.

Artières et al.[9] mention in passing the importance of the observation probabilities over the transition matrix. However, the study does not provide further information. Therefore, in section IV an experiment is presented to gain a better insight in the importance of the transition matrix. It will show that, indeed, the observation probabilities are very important. The implications of this observation and the consequences for using HMMs as a paradigm in handwriting recognition are discussed in the final section.

We will show, using generated data, that it is very difficult for the Baum-Welch algorithm to find the correct topology of the underlying Markovian process. By generating data according to a known Markov process with very specific properties (namely a left-right HMM), we know which properties the ergodic model, initially without any restrictions, should get after training. We can now show that the explicitly coded left-right topology is not found by an ergodic model. See also [10] for a discussion of the brittleness of EM algorithms.

Finally, we show that, surprisingly, removing the temporal information from an HMM does not necessarily have a large impact on performance in a real-world problem.

II. Benchmark

We will run some experiments using our own implementation as well as other HMM toolkits on a generated data set as a benchmark. It is hard to find a proper HMM benchmark for discrete, one dimensional data that has a gradual scale of increasing difficulty. The dataset that was generated for this purpose has varying degrees of symbol lexicon overlap between classes, making the completely overlapping set most difficult and the dataset with the largest between-class distance

TABLE I: Average classification performance of three randomly initialised runs on the same dataset. Please note that the standard deviation for dHMM and GHMM is 0 due to the use of a static random seed, instead of a random seed. HTK-hinit uses the `hinit` tool to initialise the model with some estimates from the data, which increases performance only slightly on these datasets. The very small difference between a separability of $\delta = 10$ and $\delta = 20$ is not visible in this table. $N_{\text{states}} = 10$, $N_{\text{symbols}} = 20$

Separability (δ)	jpHMM	dHMM	GHMM	HTK	HTK-hinit
0	1% (± 0.10)	1%	1%	1% (± 0.12)	1% (± 0.06)
1	41% (± 0.46)	40%	37%	41% (± 0.12)	41% (± 0.62)
2	66% (± 0.38)	64%	61%	66% (± 0.10)	66% (± 0.15)
3	81% (± 0.10)	78%	76%	80% (± 0.10)	80% (± 0.10)
5	95% (± 0.25)	93%	92%	94% (± 0.17)	94% (± 0.15)
10	100% (± 0.00)	100%	100%	100% (± 0.00)	100% (± 0.00)
20	100% (± 0.00)	100%	100%	100% (± 0.00)	100% (± 0.00)

least difficult. This is useful for comparing performances between runs on different feature methods, having the ability to attach a 'difficulty index' to each.

The generated data contains 100 classes, each class consists of generated transition and observation matrices. The transition matrix is a randomly[1] initialised Bakis model with $N_{\text{states}} = 10$ states, which is appropriate for variable duration modeling of left-right symbol sequences. The observation probability functions, with $N_{\text{symbols}} = 20$ symbols, are also instantiated randomly. The topology was chosen as Bakis in this benchmark. Most HMM implementations do not have restrictions on topology, except the dHMM framework (which uses a fixed, hard-coded Bakis structure). See also section III for more details on different topologies.

The gradual scale of difficulty is achieved by having multiple data sets with a varying degree of separability in symbol space. Concretely, this means that there is an overlap in lexicons between classes. A separability of δ of a dataset is defined by the following equation:

$$\begin{aligned} L_1 &= \{1 \dots N_s\} \\ L_i &= \{L_{i-1,0} + \delta \dots L_{i-1,0} + \delta + N_s\} \end{aligned} \tag{1}$$

Where δ is the separability, L_i is the lexicon, the set of symbols, to be used for class i, $L_{i,j}$ is the j^{th} element of L_i and N_s is the size of the lexicon, i.e., number of symbols per class. A separability of $\delta = 0$ is the most difficult case, because all classes share the same set of symbols: $L_1 = L_2 = \{a, b, c\}$. A separability of $\delta = 1$ means that between classes, one symbol is not re-used in the next class: $L_1 = \{a, b, c\}$ and $L_2 = \{b, c, d\}$, and so on. With more separation than symbols, a gap between the symbols is present: A dataset with $L_1 = \{a, b, c\}$ and $L_2 = \{e, f, g\}$ has a separation of $\delta = 4$.

In this section, we will show the results of running several HMM frameworks on the generated datasets. We test the popular HTK tool kit, which is well known in speech recognition[11]; GHMM, developed mainly for bioinformatics applications[12]; a framework developed by Myers and Whitson[13], dubbed dHMM here, mainly for discrete Bakis models for automatic speech recognition; and finally our own framework, developed from scratch to review in great detail the algorithmic details of HMMs, dubbed jpHMM. We also use the HTK toolkit together with the `hinit` tool to have a better initialised model, compared to random initialisation.

The benchmark datasets in this paper are all synthesized and discrete. Also, the duration of the sequences is limited. This means that the results of the current study can not directly be compared to all possible applications. However, there is no fundamental limitation on sequence length, or number of states. This can be addressed in future releases of the benchmark. For some applications and features, continuous observation modelling is beneficial[14], while for other applications and feature methods, discrete observation modelling is still very relevant[15]. In order to study the core details of HMMs, using discrete observations is interesting, since its modelling is almost trivial. Common techniques to use discrete models on continuous data are vector quantization[16], k-means clustering, or self-organizing maps (see also section IV).

The generated datasets have a separability of $\delta \in \{0, 1, 2, 3, 5, 10, 20\}$. The number of states is $N_{\text{states}} = 10$, $N_{\text{symbols}} = 20$, the length of each sequence is $|\vec{O}| = 10$ observations, yielding effectively an artificial stochastic language with 10-letter words. We have generated 100 classes with 300 sequences each. We trained models from each toolkit on all classes, and performed classification based on the most likely model for an instance.

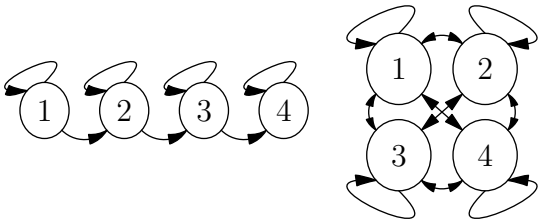
Results The classification performances on the seven datasets are reported in Table I, showing that all implementations perform roughly equally well, which is to be expected. However, we can also see the relation between benchmark difficulty, the separability δ and classification performance for five HMM implementations. From a separability of about $\delta = 5$ onward (for a dataset with $N_{\text{states}} = 10$ and $N_{\text{symbols}} = 20$) classification becomes very accurate.

III. Learning the topology of a transition matrix

In this section and the next, we describe two experiments to determine the importance of temporal modelling which is effectuated by the transition matrix in the HMM framework. The first experiment is mainly focused on the performance of the Baum-Welch algorithm, while the second shows what happens when the temporal information is removed from an HMM.

The Baum-Welch algorithm, an Expectation-Maximisation (EM) algorithm, works by initialising a model, often by using random probabilities, and then incrementally improving it. The

[1]Using the default `python` module `random`, which uses a Mersenne twister pseudorandom number generator



(a) Bakis (b) Ergodic

Fig. 1: Illustration of the Bakis and Ergodic topologies. The arrows indicate the possible transitions between the numbered states, without indicating the probability of these transitions.

initialisation step is very important due to the possibility of ending up in a local maximum, and the 'random' method is therefore very brittle, requiring human supervision.

As a first experiment to examine the transition matrix, we generate artificial data again. This has the advantage that we explicitly know the properties of the transition matrix. The specific property that we are interested in, currently, is the topology of the model. The topology is the shape of the transition matrix and there are a number of topologies possible. The most well-known is the Bakis topology, which is a left-right model that defines for each state two transition probabilities: to the current state and to the next state. Another topology is the Ergodic topology, which puts no a-priory restrictions on the transition probabilities: every state has a (possible) transition to every state (including itself). See Fig. 1 for an graphical representation of these topologies. A variant of Bakis, that has the ability to skip a state by also having a transition probability from S_i to S_{i+2}, was left out for brevity.

The experiment is set up as follows: a model is created by randomly initialising a Bakis topology with $N = 20$ states ($L = 20$ symbols). After generating 300 instances of 40 observations long with this topology, a fully Ergodic model is trained on these instances. The resulting transition matrix is examined: has it learned the fact that we used a Bakis topology to generate the training data? To be fair, we shuffle the states in the trained model to have the smallest χ^2 distance to the original model. The found hidden state S_1 in the trained model does not have to be state S_1 in the generating model, after all.

Results The original, generated model and the learned ergodic model can be visually inspected in Fig. 2. The transition matrix is converted to an image by taking the state-transition probability and coding it into a grey-scale colour: a probability of 0 is rendered as white, while a probability of 1 is rendered as black. From these figures, we can see that the Bakis topology has a diagonal structure: a probability from state S_i to S_i and to state S_{i+1}. The learned, ergodic model does not show a diagonal structure at all, even though we shuffled the matrix to have the smallest χ^2 distance to the generated Bakis model. The learned model is significantly different from the generated Bakis model ($p \ll 0.0001$, $\chi^2 = 27863$, 19 degrees of freedom), using a contingency table test on the transition frequencies[2].

From this observation we could conclude that it is difficult to learn the topology of an underlying Markov process. We performed two similar experiments to verify this finding (this time with $N = 10$ states because of compute time constraints). The first variation was done by averaging over several learned models. This is realised by generating ten Bakis models, generating 300 sequences per model and train an ergodic model on each set of sequences. The models are shuffled and averaged, and visualised in Fig. 3. Although we are not aware of this extensive procedure being done in the literature, it appears to be useful to see whether a diagonal pattern can be found, on average, even when it is difficult to see in a single model.

It is well known that one requires a large set of training sequences to estimate the right model. We used this idea in another method of trying to find the underlying Bakis structure using an ergodic model. Instead of averaging over ten models, we now use ten times as much data. This gives the training algorithm more data to learn the structure from. Fig. 4(b) shows the results of estimating the topology from 3000 sequences, that were generated using the model shown in Fig. 4(a).

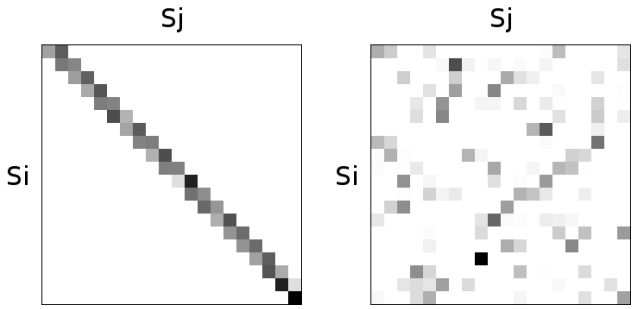
Both Fig. 3 and 4 show that trying really hard to force an ergodic model to find the Bakis structure can result in a slight tendency towards a diagonal structure under highly artificial training conditions. The desired diagonal probabilities are present in the learned ergodic models, but the off-diagonal probabilities are abundant in these models as well. From the diagonals, the self-recurrent state-transition probabilities are most pronounced. This shows that it may be very difficult to find the underlying structure of a Markov process using the Baum-Welch algorithm (given the specific parameters). From this and pilot studies, we conclude that it is less difficult to find a diagonal structure for $N = 10$ states than for $N = 20$ (which is more common). We will verify this in a future study.

IV. The importance of temporal modelling

We are also interested in what happens when we remove the temporal information from the transition matrix. This means that we create a *flat* topology: all transition probabilities are equally probable: $a_{ij} = \frac{1}{N}$, where N is the number of states. During training, the transition matrix will continuously be made uniform (i.e., flat) in each iteration. This is necessary because the observation probabilities may no longer be correct when adjusting the transition probabilities after training. The flat topology can be viewed as an orderless "bag of states". We will now compare how well models with this topology compare to models with a Bakis or ergodic topology.

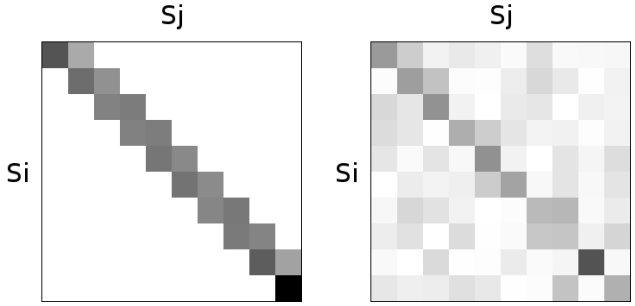
In this experiment we train an HMM on discrete features, extracted from handwritten word images. The dataset uses a single handwritten book from the collection of the Dutch National Archives[18]. We use two features: fragmented connected component contours (FCO^3) and a sliding window,

[2]The Kolmogorov-Smirnov test cannot be used since there is no meaningful univariate axis to integrate the probabilities[17].



(a) Target (Bakis) model (b) Learned (ergodic) model

Fig. 2: Transition probability matrices. After generating a model of $N = 20$ states, Fig. 2(a), 300 sequences were generated with this model. A new model was trained on this data, and after shuffling the learned model such that it is closest to the original model, we can see that it has not learned the topology, Fig. 2(b). A probability of 0 is rendered as white, a probability of 1 as black. χ^2 distance $= 48$



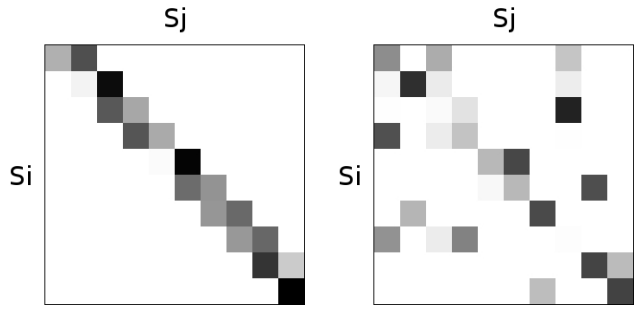
(a) Target (Bakis) model (b) Learned (ergodic) model

Fig. 3: After generating ten models with the number of states reduced to $N = 10$, per model 300 sequences were generated, otherwise similar to Fig. 2. New models were trained on each of these 300 sequences and the models were averaged. Fig. 3(a) shows the average model of the generated models, while 3(b) shows the average learned model, with a vague tendency towards diagonal state-transitions, mostly the self-recurrent transitions, while the next-state-transitions show a less pronounced pattern. Average χ^2 distance $= 16$

both quantized using a Kohonen self-organizing feature map (SOFM)[19].

For the FCO^3 feature, the image is broken up into a sequence of fragmented connected component contours[20]. Each of these contours is then quantized into a discrete index of a SOFM of 70×70 nodes. This means the lexicon consists of 4900 symbols. We have selected 130 classes with at least 51 training instances, with a total of 30 869 labelled instances. Because the average length of the words was 4.4 FCO^3 observations, the number of states was chosen to be 3.

The second feature is extracted using a sliding window of 4 by 51 pixels, centered around the centroid of black pixels over the entire word zone. The SOFM for this feature, with 25×25 nodes, was a lot smaller than the FCO^3 feature map, due to time constraints. Centering around the centroid with a height of 51 pixels means that the outstanding sticks and (partial) loops of ascenders and descenders are still preserved, while reducing the size of the image considerably. We limited the number of classes in the experiments with this feature to



(a) Target (Bakis) model (b) Learned (ergodic) model

Fig. 4: Instead of averaging over ten models as in Fig. 3, we now use ten times as much generated sequences to train a single model (3000 sequences). We see that there is a small tendency towards diagonal (Bakis-like) state transitions, but it is not very strong. χ^2 distance between the two distributions $= 14$

20, with a total of 4 928 labelled instances. The average length of observation sequences for the sliding window feature was 65.9 observations, which led us to use $N = 27$ states[3].

For classification, an HMM λ is first trained on the instances of each class, and then classification can be performed using $\text{argmax}_{\lambda \in \Lambda}[\log P(O|\lambda)]$, where Λ is the set of all trained models and O is the test sequence. To investigate the role of the state-transition probabilities, we perform the experiments with three topologies: Bakis, ergodic and flat, which is the topology where all transition probabilities are equally probable. We perform the experiments on both features using 7-fold cross validation, with our own implementation, jpHMM.

Results The results are summarised in Table II and III. We can see that the results of classification with the FCO^3 feature are very close together (and not statistically significant, with ANOVA, $p > 0.05$). There is a significant difference in the classification performance using the sliding window feature (ANOVA, $p < 0.001$), but the drop in performance is not as dramatic as would be expected from the removal of temporal information in the Markov paradigm.

Please note that the HMMs were used as a measurement tool to find differences between transition models. They have an average performance, avoiding ceiling effects. Also, the FCO^3 feature is a feature developed for writer identification, not handwriting recognition per-se. The sliding window feature could be fine-tuned further by changing the size of the Kohonen map, the window, the number of states, etc. In this experiment we are interested in evaluating HMM topologies, not in maximising the recognition performance.

V. Discussion

We set out to reevaluate hidden Markov models, by creating a benchmark for discrete HMMs, and running experiments to investigate the importance of the transition matrix.

Using the benchmark, we found that for discrete observations, all common HMM tools have similar performances.

[3]The increase in number of states is most likely the reason for the increased time necessary for training

TABLE II: Results of the FCO^3 experiment. Performances reported are averages over 7 folds, with 130 classes and at least 51 instances per class in the training set. As can be seen, all topologies perform around 60%. Flat models do not perform significantly worse.

Topology	Classification performance
Bakis	59.9% ± 0.9
Ergodic	59.5% ± 0.9
Flat	59.1% ± 0.8

TABLE III: Results of the sliding window experiment. Performances reported are averages over 7 folds, with 20 classes and at least 51 instances per class in the training set. Differences between the topologies are statistically significant ($p < 0.001$) although the difference between the flat and ergodic topologies is not as dramatic as expected

Topology	Classification performance
Bakis	75.2% ± 2.0
Ergodic	78.5% ± 1.2
Flat	71.1% ± 1.3

Furthermore, we can now measure the difficulty of discrete data, by comparing the performances of discrete HMMs with the performances of the benchmarks with different degrees of difficulty. In the future we want to extend the current study with continuous density HMMs as well.

While it is barely presented in the literature, the fact that the transition matrix is of a smaller importance than the observation probabilities is well known from personal communications at, e.g., conferences. We have done two experiments to establish the importance of the transition matrix, and found that indeed the observation probabilities have a large impact on recognition performance. The results of these experiments showed that (a) it is hard to learn the correct, known topology of the underlying Markov process and (b) that classification with the temporal information removed from the HMMs can also result in reasonably performant classifiers.

Regarding (a), it appears that the Baum-Welch training method is not very reliable to estimate the underlying transition structure in the data. As noted in [10], EM is brittle and very sensitive to the initialisation process. We have shown that the Baum-Welch method was unable to find the Bakis topology from generated data when initialised as a full Ergodic model. We have previously studied initialisation of models to prevent local maxima[21], but this still requires a lot of human modelling effort, specifically for each problem variant.

Regarding our finding (b), that classification with temporal information removed can result in performant classifiers, we believe that the observation probabilities are very important. This supports our view that the quality of the features may require more attention than the aspect of temporal modelling. From a more scientific point of view, it is still a challenge to adapt the Baum-Welch estimation algorithm to correctly estimate the Markov parameters of an observed sequential process.

Even though these findings expose limitations of HMM and its training procedure, the fact that recognition performance is not degraded dramatically when removing temporal information from HMMs implies that dynamic programming (i.e., the operational stage) is a strong principle. Also, the Markov assumption remains appealing from a theoretical perspective.

Given these considerations, we feel 1) that it may be misleading to stress the *hidden* aspect of HMMs, because of the relatively minor role the hidden states play in achieving good performance, 2) the Baum-Welch algorithm should be replaced with a less brittle method, and 3) although the HMM principles mentioned above are strong, there are many tricks of the trade, that are not treated well in literature (see also the Appendix).

REFERENCES

[1] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[2] S. R. Eddy, "Profile hidden Markov models." *Bioinformatics*, vol. 14, no. 9, pp. 755–763, 1998.

[3] H. Bunke, M. Roth, and E. G. Schukat-Talamazzini, "Off-line cursive handwriting recognition using hidden Markov models," *Pattern recognition*, vol. 28, no. 9, pp. 1399–1413, 1995.

[4] T. Plötz and G. A. Fink, "Markov models for offline handwriting recognition: a survey," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 12, no. 4, pp. 269–298, 2009.

[5] S.-s. Kuo and O. E. Agazzi, "Machine vision for keyword spotting using pseudo 2D hidden Markov models," in *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, vol. 5. IEEE, 1993, pp. 81–84.

[6] H.-S. Park and S.-W. Lee, "A truly 2-D hidden Markov model for off-line handwritten character recognition," *Pattern Recognition*, vol. 31, no. 12, pp. 1849–1864, 1998.

[7] A. Benouareth, A. Ennaji, and M. Sellami, "Semi-continuous HMMs with explicit state duration for unconstrained Arabic word modeling and recognition," *Pattern Recognition Letters*, vol. 29, no. 12, pp. 1742–1752, 2008.

[8] V. Borkar, K. Deshmukh, and S. Sarawagi, "Automatic segmentation of text into structured records," in *ACM SIGMOD Record*, vol. 30, no. 2. ACM, 2001, pp. 175–186.

[9] T. Artières, P. Gallinari, H. Li, S. Marukatat, and B. Dorizzi, "From character to sentences: A hybrid Neuro-Markovian system for on-line handwriting recognition," *Series in Machine Perception and Artificial Intelligence*, vol. 47, pp. 145–170, 2002.

[10] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 3, pp. 381–396, 2002.

[11] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland, *The HTK Book, version 3.4*. Cambridge, UK: Cambridge University Engineering Department, 2006.

[12] "The General Hidden Markov Model Library (GHMM)," http://ghmm.org, 2003, accessed February 22, 2014.

[13] R. Myers and J. Whitson, "HMM: Hidden Markov Model software for automatic speech recognition," https://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/speech/systems/hmm/0.html, 1994, accessed February 22, 2014.

[14] F. R. Chen, L. D. Wilcox, and D. S. Bloomberg, "A comparison of discrete and continuous hidden Markov models for phrase spotting in text images," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1. IEEE, 1995, pp. 398–402.

[15] G. Rigoll, A. Kosmala, J. Rattland, and C. Neukirchen, "A comparison between continuous and discrete density hidden Markov models for cursive handwriting recognition," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 2. IEEE, 1996, pp. 205–209.

[16] J. Schenk, S. Schwärzler, G. Ruske, and G. Rigoll, "Novel VQ designs for discrete HMM on-line handwritten whiteboard note recognition," in *Pattern Recognition*. Springer, 2008, pp. 234–243.

[17] G. Babu and E. D. Feigelson, "Astrostatistics: Goodness-of-fit and all that!" in *Astronomical Data Analysis Software and Systems XV*, vol. 351, 2006, p. 127.

[18] T. Van der Zant, L. Schomaker, and K. Haak, "Handwritten-word spotting using biologically inspired features," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 11, pp. 1945–1957, 2008.
[19] T. Kohonen, "Adaptive, associative, and self-organizing functions in neural computing," *Applied Optics*, vol. 26, no. 23, pp. 4910–4918, 1987.
[20] L. Schomaker, K. Franke, and M. Bulacu, "Using codebooks of fragmented connected-component contours in forensic and historic writer identification," *Pattern Recognition Letters*, vol. 28, no. 6, pp. 719–727, 2007.
[21] T. K. Bhowmik, J.-P. van Oosten, and L. Schomaker, "Segmental K-means learning with mixture distribution for HMM based handwriting recognition," in *Pattern Recognition and Machine Intelligence*. Springer, 2011, pp. 432–439.

APPENDIX

Implementing an HMM framework from scratch is not trivial. The canonical paper by Rabiner[1] contains all the theory necessary, but it may require some extra considerations to make implementation easier. We will give some of these considerations here. The approach used for implementing jpHMM is taken in part from A. Rahimi (2000)[4].

Scaling forward and backward variables

The first issue to address is scaling the forward and backward variables $\alpha_t(j)$ and $\beta_t(j)$. The forward variable is the probability of the partial observation sequence up to time t and being in state S_j at time t, given the model λ: $\alpha_t(j) = P(O_1 O_2 \cdots O_t, q_t = S_j|\lambda)$. The backward variable is the probability of the partial observation sequence from time $t+1$ to time T, given state S_j at time t and the model λ: $\beta_t(j) = P(O_{t+1} O_{t+2} \cdots O_T|q_t = S_j, \lambda)$.

These variables need to be scaled to avoid problems with floating point representations in code. Since the forward variable $\alpha_t(j)$ usually consists of many products of transition and observation probabilities, they tend to approach 0 quickly. On a computer, these variables are bound by a finite precision floating point representation.

A scaling can be applied to both $\alpha_t(j)$ and $\beta_t(j)$, to keep the calculations in range of a floating point representation. Rabiner proposes to use the scaling factor $c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}$, which is independent of state. This means that $\sum_{i=1}^{N} \hat{\alpha}_t(i) = 1$. Both $\alpha_t(j)$ and $\beta_t(j)$ are scaled with the same factor, c_t.

The recursion formulae defined by Rabiner are theoretically correct, but hard to use for implementation because it is unclear that one needs to use the scaled $\hat{\alpha}_t(i)$ in the computation for c_{t+1}. Rahimi therefore proposes the following computation steps:

$$\begin{aligned}
\overline{\alpha}_1(i) &= \alpha_1(i) \\
\overline{\alpha}_{t+1}(j) &= \sum_{i=1}^{N} \hat{\alpha}_t(i) a_{ij} b_j(O_{t+1}) \\
c_{t+1} &= \frac{1}{\sum_{i=1}^{N} \overline{\alpha}_{t+1}(i)} \\
\hat{\alpha}_{t+1}(i) &= c_{t+1} \overline{\alpha}_{t+1}(i)
\end{aligned}$$

[4]Please find Rahimi's solution at http://alumni.media.mit.edu/~rahimi/rabiner/rabiner-errata/rabiner-errata.html, accessed January 23, 2014.

Rabiner leaves out the full steps to compute $\hat{\beta}_t(i)$. We can use the following (also from Rahimi):

$$\begin{aligned}
\overline{\beta}_T(i) &= \beta_T(i) \\
\overline{\beta}_t(j) &= \sum_{i=1}^{N} a_{ij} b_j(O_{t+1}) \hat{\beta}_{t+1}(i) \\
\hat{\beta}_t(i) &= c_t \overline{\beta}_t(i)
\end{aligned}$$

We can express the probability of a sequence given a model using $P(\mathbf{O}|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t}$, but since this is also a product of probabilities, we are better off using the sum of log probabilities: $\log[P(\mathbf{O}|\lambda)] = -\sum_{t=1}^{T} \log c_t$.

Multiple observation sequences of variable duration

While implementing the reestimation formulae for multiple observation sequences of variable duration, we ran into the problem of requiring $P(\mathbf{O}^{(k)}|\lambda)$, where $\mathbf{O}^{(k)}$ is the kth observation sequence. We can no longer compute this, because we now use log-probabilities. However, we can rewrite these formula to no longer use $P(\mathbf{O}^{(k)}|\lambda)$. The full derivations are left out, but are essentially the same as those by Rahimi. We will also show the reestimation formula for π, because both Rabiner and Rahimi do not mention it. They assume a strict left-right model, such as Bakis, where $\pi_1 = 1$ and $\pi_i = 0$ for $i \neq 1$.

We will use the following equalities:

$$\begin{aligned}
\prod_{s=1}^{t} c_s^k &= C_t^k \\
\prod_{s=t+1}^{T_k} c_s^k &= D_{t+1}^k \\
\prod_{s=1}^{T_k} c_s^k &= C_t^k D_{t+1}^k = C_{T_k}^k \\
\frac{1}{\prod_{t=1}^{T_k} c_t^k} &= \frac{1}{C_{T_k}^k} = P(\mathbf{O}^{(k)}|\lambda)
\end{aligned}$$

where c_t^k is the scaling factor $\frac{1}{\sum_{j=1}^{N} \overline{\alpha}_t^k(j)}$. Because we now have a new way of representing $P(\mathbf{O}^{(k)}|\lambda)$ as $\frac{1}{C_{T_k}^k}$, we can substitute that into the reestimation equations, leading to the following equations after some rewriting:

$$\begin{aligned}
\overline{a}_{ij} &= \frac{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(i) a_{ij} b_j(O_{t+1}^{(k)}) \hat{\beta}_{t+1}^k(j)}{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(i) \hat{\beta}_t^k(i) \frac{1}{c_t^k}} \\
\overline{b}_j(\ell) &= \frac{\sum_{k=1}^{K} \sum_{t \in [1,T_k-1] \wedge O_t = v_\ell} \hat{\alpha}_t^k(i) \hat{\beta}_t^k(i) \frac{1}{c_t^k}}{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(j) \hat{\beta}_t^k(j) \frac{1}{c_t^k}} \\
\overline{\pi}_i &= \frac{\sum_{k=1}^{K} \hat{\alpha}_1^k(i) \hat{\beta}_1^k(i) \frac{1}{c_1^k}}{\sum_{j=1}^{N} \sum_{k=1}^{K} \hat{\alpha}_1^k(j) \hat{\beta}_1^k(j) \frac{1}{c_1^k}}
\end{aligned}$$

For the full details and derivations of the reestimation equations, please see the explication by Rahimi or contact the authors of this study. The documented code for jpHMM will be published on-line soon.